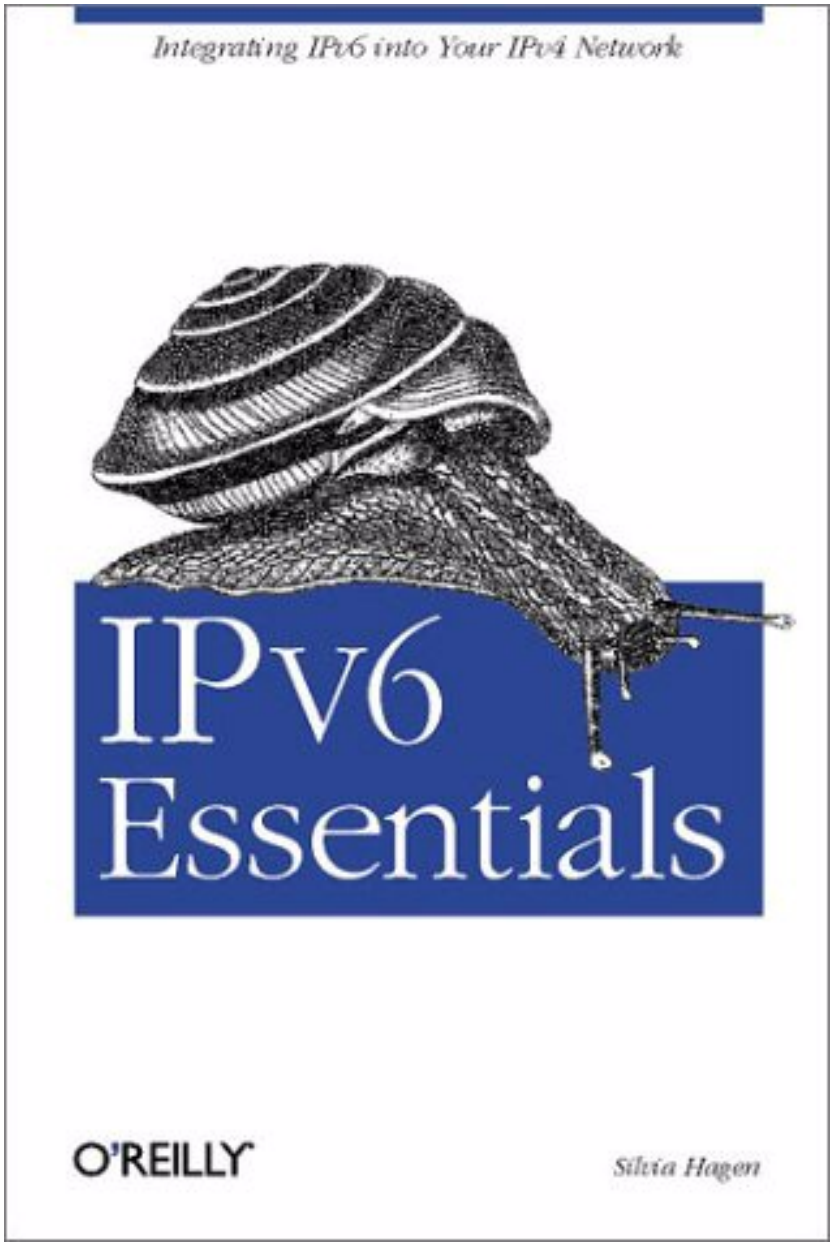


ISBN: 0-596-00125-8

Table of Contents:

Chapter 1. IPv6 Versus IPv4

IPv6 is sometimes called the Next Generation Internet Protocol, or IPng. Even though the Internet is seen as a relatively new technology, the protocols and technologies that make it work were developed in the 1970s and 1980s. The current Internet and all our corporate and private intranets use IPv4. Now, with IPv6, the first major upgrade of the Internet protocol suite is on the horizon or maybe even closer. Close enough, anyway, to start taking it seriously.

1.1 The History of IPv6

The effort to develop a successor protocol to IPv4 was started in the early 1990s by the Internet Engineering Task Force (IETF). Several parallel efforts began simultaneously, all trying to solve the foreseen address space limitation as well as provide additional functionality. The IETF started the IPng area in 1993 to investigate the different proposals and to make recommendations for further procedures.

The IPng area directors of the IETF recommended the creation of IPv6 at the Toronto IETF meeting in 1994. Their recommendation is specified in RFC 1752, "The Recommendation for the IP Next Generation Protocol." The Directors formed an Address Lifetime Expectation (ALE) working group, whose job was to determine whether the expected lifetime for IPv4 would allow the development of a protocol with new functionality or if the remaining time would only allow for developing an address space solution. In 1994, the ALE working group projected the IPv4 address exhaustion to occur sometime between 2005 and 2011, based on the statistics that were available at that time.

For those of you who are interested in the different proposals, here's some more information about it (from RFC 1752). There were four main proposals called CNAT, IP Encaps, Nimrod, and Simple CLNP. Three more proposals followed: the P Internet Protocol (PIP), the Simple Internet Protocol (SIP), and TP/IX. After the March 1992 San Diego IETF meeting, Simple CLNP evolved into TCP and UDP with Bigger Addresses (TUBA) and IP Encaps evolved into IP Address Encapsulation (IPAE). IPAE merged with PIP and SIP and called itself Simple Internet Protocol Plus (SIPP). The TP/IX working group changed its name to Common Architecture for the Internet (CATNIP). The main proposals were now CATNIP, TUBA, and SIPP. For a short discussion of the proposals, refer to RFC 1752.

CATNIP is specified in RFC 1707, TUBA in RFC 1347, RFC 1526, and RFC 1561, and SIPP in RFC 1710.

The Internet Engineering Steering Group approved the IPv6 recommendation and drafted a Proposed Standard on November 17, 1994. The core set of IPv6 protocols became an IETF Draft Standard on August 10, 1998.

Why is the new protocol not IPv5? The version number 5 could not be used because it had been allocated to an experimental stream protocol.

1.2 Overview of Functionality

IPv6 is one of the most significant network and technology upgrades in history. It will slowly grow into your existing IPv4 infrastructure and positively impact your network. Reading this book will prepare you for the next step of networking technology evolution. IPv6 product development and implementation efforts are already underway all over the world. IPv6 is designed as an evolutionary step from IPv4. It is a natural increment to IPv4, can be installed as a normal software upgrade in most Internet devices, and is

interoperable with the current IPv4. IPv6 is designed to run well on high performance networks like Gigabit Ethernet, ATM, and others, as well as low bandwidth networks (e.g., wireless). In addition, it provides a platform for new Internet functionality that will be required in the near future, such as extended addressing, better security, and quality of service (QoS) features.

IPv6 includes transition and interoperability mechanisms that are designed to allow users to adopt and deploy IPv6 step by step as needed and to provide direct interoperability between IPv4 and IPv6 hosts. The transition to a new version of the Internet Protocol (IP) must be incremental, with few or no critical interdependencies, if it is to succeed. The IPv6 transition allows users to upgrade their hosts to IPv6 and network operators to deploy IPv6 in routers with very little coordination between the two groups.

The main changes from IPv4 to IPv6 can be summarized as follows:

Expanded addressing capability and autoconfiguration mechanisms

The address size for IPv6 has been increased to 128 bits. This solves the problem of the limited address space of IPv4 and offers a deeper addressing hierarchy and simpler configuration. There will come a day when you will hardly remember how it felt to have only 32 bits in an IP address. Network administrators will love the autoconfiguration mechanisms built into the protocol. Multicast routing has been improved, with the multicast address being extended by a scope field. And a new address type has been introduced, called Anycast address, which can send a message to the nearest single member of a group.

Simplification of the header format

The IPv6 header has a fixed length of 40 bytes. This actually accommodates only an 8-byte header plus two 16-byte IP addresses (source and destination address). Some fields of the IPv4 header have been removed or become optional. This way, packets can be handled faster with lower processing costs.

Improved support for extensions and options

With IPv4, options were integrated into the basic IPv4 header. With IPv6, they are handled as *Extension headers*. Extension headers are optional and only inserted between the IPv6 header and the payload, if necessary. This way the IPv6 packet can be built very flexible and streamlined. Forwarding IPv6 packets is much more efficient. New options that will be defined in the future can be integrated easily.

Extensions for authentication and privacy

Support for authentication, and extensions for data integrity and data confidentiality, have been specified and are inherent.

Flow labeling capability

Packets belonging to the same traffic flow, requiring special handling or quality of service, can be labeled by the sender. Real-time service is an example where this would be used.

For a current list of the standardization status of IPv6, you can refer to http://playground.sun.com/pub/ipng/html/specs/standards.html.

1.3 Transition Aspects

Is IPv6 worth all the migration and upgrade headaches? Will it ever become the IP of the future? Can't IPv4 extensions offer all that functionality? After all, we have Network Address Translation (NAT) to solve address space problems and IPSEC to provide security.

The 128-bit address space is the most obvious feature of the new protocol, but it is not the only important change. The IPv6 package includes important features such as higher scalability, better data integrity, QoS features, autoconfiguration mechanisms that make it manageable even for high numbers of dynamically connecting devices, improved routing aggregation in the backbone, and improved multicast routing.

Extensions for IPv4 that have been widely deployed, such as NAT, should be viewed as good solutions but only for limited short-term scenarios. In the long term, nothing can replace IPv6's features for inherent secure end-to-end connectivity. Multimedia and interactive, transaction-oriented network applications require high levels of connectivity that can only be provided by IPv6. In the future, an unforeseeable number of new devices may want to connect to our networks, including devices such as Personal Digital Assistants (PDAs), mobile phones, smart set-top boxes with integrated web browsers, home entertainment systems, coffee machines, refrigerators, and car devices. The list is endless. Only IPv6, with its extended address space and advanced autoconfiguration and mobility features, can manage such devices. There is no comparable alternative technology in sight.

1.4 IPv6 Alive

There are already a surprising number of global test networks and even commercial networks running over IPv6. I discuss some interesting examples in the next sections. In order to describe what they are doing, I use some IPv6-specific terms that are probably not familiar to you yet. They are all explained in this book.

In February 2002 over 120 production networks have been allocated IPv6 address prefixes. For a current list, refer to http://www.dfn.de/service/ipv6/ipv6aggis.html.

1.4.1 The 6Bone

The 6Bone started out as a network of IPv6 islands working over the existing IPv4 infrastructure of the Internet by tunneling IPv6 packets in IPv4 packets. The tunnels were mainly statically configured point-to-point links. The 6Bone became a reality in early 1996 as a result of an initiative of several research institutes. The first tunnels were established between the IPv6 laboratories of G6 in France, UNI-C in Denmark, and WIDE in Japan.

1.4.1.1 Structure of the 6Bone

The 6Bone is structured as a hierarchical network of two or more layers. The top layer consists of a set of backbone transit providers, called pseudo Top Level Aggregators (pTLAs), which use BGP4+ as a routing protocol. The bottom layer is comprised of leaf sites connected via the 6Bone. Zero or more intermediate layers, called pseudo Next Level Aggregators (pNLAs), interconnect leaf sites and the pTLA backbone networks.

1.4.1.2 Addressing

IPv6 unicast addressing of node interfaces (for both end systems and routers) is based on RFC 2374, which covers the Aggregatable Global Unicast address format. 6Bone backbone networks play the role of experimental TLAs, called pseudo TLAs (pTLAs), and assign address space to pseudo NLAs (pNLAs) and leaf sites. The prefix assigned to the 6Bone is `3ffe::/16` (RFC 2471). These pTLA backbone networks

are currently allocated 32-bit prefixes (previously, 24- and 28-bit prefixes were allocated) that must be administered according to the rules defined for pTLAs. So every pTLA plays the role of an experimental top-level ISP and assigns chunks of its addressing space to directly connected transit and leaf sites without breaking aggregation inside the 6Bone backbone.

1.4.1.3 Growth

The 6Bone is growing fast. In December 1997 there were 43 backbone sites and 203 leaf sites registered. In December 1998 there were 51 backbone sites and 332 leaf sites. In January 2000 there were 67 backbone sites and 505 leaf sites.

I gave up on trying to find a nice picture of the world with the 6Bone backbone sites on it. The 6Bone has grown too big to display it in one screenshot. If you want to get a feeling for the size and workings of the 6Bone, go to http://www.cs-ipv6.lancs.ac.uk/ipv6/6Bone and look at the maps, statistics, and tools.

At the time of this writing, the number of nodes in the 6Bone has just reached 1000 nodes and grows daily. Find an updated list at http://www.cs-ipv6.lancs.ac.uk/ipv6/6Bone/Whois/index.html#full.

1.4.1.4 Joining the 6Bone

Membership in the 6Bone is open to anyone. Reasons for joining, besides the fun of it, would be to gain early experience working with IPv6, to build the expertise necessary to make decisions about when and how to use IPv6 for production networks, and to have working access to IPv6 servers and resources. Joining the 6Bone connects you with a cool crowd of people who want to be on top of technology and are willing to share their experience.

The 6Bone community spans the globe and is very active and enthusiastic. By joining, you not only gain access to the network and the common experience of those in it; you can also participate and help develop protocols, programs, and procedures.

If you are interested in joining the 6Bone, here's the link: http://www.6bone.net/6bone_hookup.html.

There are different ways for you to connect to either the 6Bone or production IPv6 networks:

- Become an end site of an existing 6Bone ISP (which means you will get your 48-bit IPv6 external routing prefix from that ISP's TLA). You can also get temporary address allocations from tunnel broker sites (see the 6Bone home page for more information).
- Apply for your own 6Bone TLA (if you are an ISP) based on the 6Bone process.
- To get your first production IPv6 address, find a production IPv6 ISP (i.e., an ISP that has a sub-TLA) from which to get your prefix. Note that you can partially qualify for a sub-TLA production prefix if you have a 6Bone *pTLA* prefix (at least during the early phase of production prefix allocation).
- Use the "6to4" automatic tunneling mechanism. This allows you to specify the IPv4 address of your end user site router for an IPv6-over-IPv4 tunnel to reach your end user site. Addresses of this type have the first 16 bits of `2002::/16`, with the next 32 bits containing the IPv4 address of a router on your site supporting this mechanism (thus making up the entire 48-bit external routing prefix). Refer to Chapter 10 for more information on the "6to4" automatic tunneling mechanism.

Now all you really need is a router and a host running IPv6 stacks. Almost all router vendors have either production stacks or beta stacks available. Refer to http://playground.sun.com/pub/ipng/html/ipng-implementations.html for a list of router and host implementations.

Obviously you need an entry point into the 6Bone. Try to find one that is close to your normal IPv4 path into the Internet. You can find a good 6Bone TLA on the 6Bone home page at http://www.6bone.net/6bone_pTLA_list.html. Use *traceroute* to determine the closest path.

1.4.2 IPv6 Commercial Networks

Since I started writing this book, a lot has happened in the development of IPv6. There are many production networks worldwide that have already been assigned IPv6 address prefixes. We picked four examples of companies that made their step into the future by offering IPv6 services.

1.4.2.1 vBNS+

vBNS+ is a specialized US IP network that supports high-performance, high-bandwidth applications. The vBNS+ network supports both native IPv6-over-ATM connections and tunneled IPv6-in-IPv4 connections. The vBNS+ service has been assigned its own sTLA from ARIN, as well as a pTLA for the 6Bone, and is delegating address space under these assignments to vBNS-attached sites. For more information, refer to their site at http://www.vbns.net.

1.4.2.2 Telia Sweden

In summer 2001, Telia, in Sweden, announced its intention to build a new generation Internet based on IPv6. By the end of 2001, connection points were installed in Stockholm, Farsta, Malmoe, Gothenburg (Sweden), Vasa (Finland), Oslo, Copenhagen, and London.

I spoke with the project manager at Telia because I thought that his early adopter input might be interesting for companies that consider going into IPv6. Telia's intent was to break through the lethargy of the chicken and the egg problem: vendors do not develop because the market is not asking for it, and the market doesn't ask for it because vendors don't develop. So Telia made the decision to create a market by building an IPv6 network and opening it to the public. Telia's hope is that, through the publicity of its endeavor, other companies will follow suit, and the acceptance and development of IPv6 will increase.

At the current stage of its rollout, Telia is keeping the IPv6 network separate from the existing IPv4 infrastructure. There were different reasons for this decision:

- It was easier to start by keeping the networks separate. Telia does not have to educate all of its IPv4 engineers to use IPv6 overnight.
- If there are problems with the IPv6 network, the IPv4 network is not affected in any way.
- It is less complex to configure if the networks are separate.

The new network is primarily built as a native IPv6 network. In some instances, tunnels over IPv4 are used. Currently, Telia is offering an IPv6 transport service to a limited number of customers. It will add features and gradually open the IPv6 network as a general service for everyone. Telia uses Hitachi routers that support IPv6 in hardware (versus software implementations).

After rolling out the first connection points, Telia concluded that market support for IPv6 was sufficient to get started. There are applications that will need to be ported to IPv6, but Telia recommends that companies and ISPs start right away. The foundation is here and when IPv6 is implemented on a broader range, vendors and application developers will be encouraged to speed up development.

1.4.2.3 Internet Initiative Japan

Another company that offers IPv6 transport services is Internet Initiative Japan (IIJ), Japan's leading Internet access and solutions provider, which targets high-end corporate customers. IIJ offers a trial IPv6 service (tunneling through IPv4) and a native IPv6 service that is independent from existing IPv4

networks. In December 2001 IIJ extended its IPv6 services to individual users connecting through IIJmio DSL/SF, an ADSL Internet service.

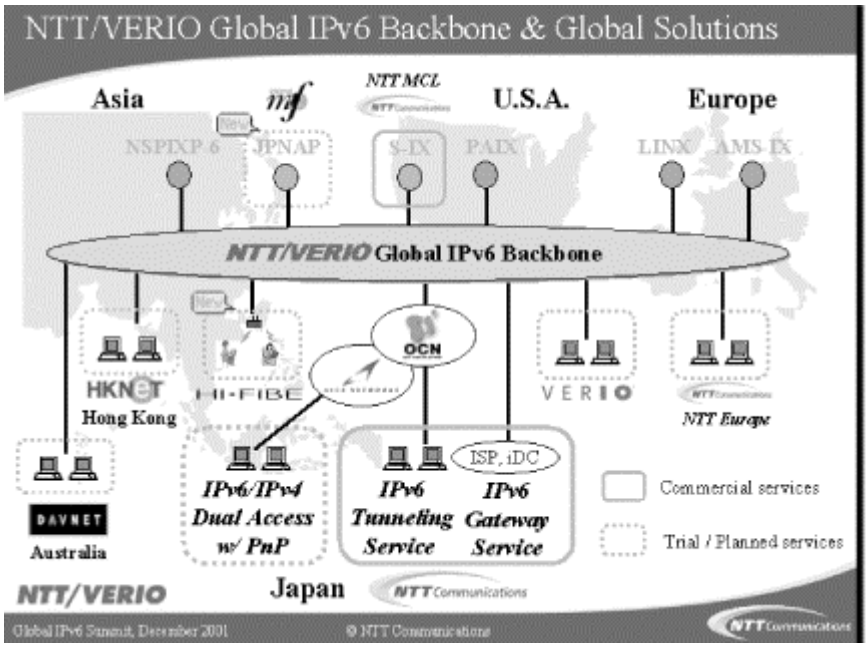
For information about IIJ's services, refer to http://www.iij.ad.jp/IPv6/index-e.html.

1.4.2.4 NTT Communications Corporation

NTT Laboratories started one of the largest global IPv6 research networks in 1996. Trials of their global IPv6 network, using official IPv6 addresses, began in December 1999. Since spring 2001, NTT Communications has offered commercial IPv6 services.

In April 2001 the company started their commercial IPv6 Gateway Service. This native IPv6 backbone service connects sites in Japan to the NTT/VERIO Global Tier1 IPv6 backbone deployed over Asia, the U.S., and Europe. Monitoring and operation continues 7 days a week, 24 hours a day, through NTT Communications NOC in Tokyo, Japan and Verio NOC in Dallas, US. Figure 1-1 shows the layout of the backbone.

Figure 1-1. NTT/VERIO's global IPv6 backbone



The IPv6 Gateway Service offers native IPv6 transport. Also shown on the picture is the IPv6 Tunneling Service that NTT has offered since June 2001. It uses the existing IPv4 network to enable NTT's partners to access the IPv6 network, using IPv6-over-IPv4 tunneling techniques via dedicated lines. The newest addition is the IPv6/IPv4 Dual Access point with plug-and-play functionality, which became available in the first quarter of 2002. It is shown in dotted lines on Figure 1-1. The first customers to use the native backbone service were BIGLOBE/NEC Corporation, CHITA MEDIAS NETWORK INC., Toshiba, InfoSphere/NTTPC Communications, Fujitsu Matsushita Graphic Communication Systems, Inc., and MEX/Media Exchange, Inc. In June 2001, NTT demonstrated applications running over IPv6, including a remote control camera running over IPv6 and videoconferencing using IPv6.

The routing protocols used are BGP4+ and RIPng, IS-IS (which will be on the global backbone in the near future), and OSPFv3 (which is used at NTT's Japan domestic backbone). What NTT lacked was ICMPv6

polling in commercial operational tools. They utilize their own custom-developed router configuration tools and network management tools that support IPv6.

NTT offers Points Of Presence (POPs) all over the world, currently in London, Palo Alto, San Jose, Seattle, and Tokyo. They plan to extend their services throughout the world; the next POPs will be in Hong Kong and Australia. NTT's services include official IPv6 addresses from their *sTLA* block, IPv6 Internet connectivity, and DNS reverse zone delegation for the subscriber's IPv6 address space.

For an overview of NTT's global IPv6 services and how you can participate and connect, refer to http://www.v6.ntt.net/globe/index-e.html.

1.4.3 Links to Other IPv6 Networks

There are a large number of international IPv6 test and research networks. You can find some interesting links in the following list:

The 6Ren

The 6Ren is a voluntary coordination initiative of research and education networks that provide production IPv6 transit service to facilitate high-quality, high-performance, and operationally robust IPv6 networks. Participation is free and open to all research and education networks that provide IPv6 service. Other profit and nonprofit IPv6 networks are also encouraged to participate. The 6Ren web site can be found at http://www.6ren.net.

The 6Net

The 6Net is a high-capacity IPv6 research network coordinated by Cisco, with more than 30 members. Their home page can be found at http://www.sixnet.org.

DRENv6

The Defense Research and Engineering Network (DREN) is a major component of the DoD High Performance Computing Modernization Program (HPCMP). Its purpose is to provide high-performance network connectivity to various communities of interest in the DoD, including research and development, modeling and simulation, and testing and evaluation. DREN also provides connectivity to other high-performance backbones and Federal networks to serve the needs of these communities. DREN is also a research network; it provides a test bed for testing new protocols and applications. DREN provides both ATM cell-based services and IP frame-based services. The DREN IPv6 network is one of the services provided as part of DREN. The DREN web site is at http://www.v6.dren.net.

Chapter 2. The Structure of the IPv6 Protocol

This chapter explains the structure of the IPv6 header and compares it to the IPv4 header. It also discusses Extension headers, which are new in IPv6.

The header structure of an IPv6 packet is specified in RFC 2460. The header has a fixed length of 40 bytes. The two fields for source and destination addresses each use 16 bytes (128 bits), so there are only 8 bytes for general header information.

2.1 General Header Structure

In IPv6, five fields from the IPv4 header have been removed:

- Header Length
- Identification
- Flags
- Fragment Offset
- Header Checksum

The Header Length field was removed because it is not needed in a header with a fixed length. In IPv4 the minimum header length is 20 bytes, but if options are added, it can be extended in 4-byte increments up to 60 bytes. Therefore, with IPv4, the information about the total length of the header is important. In IPv6 options are defined by Extension headers (covered later in this chapter).

The Identification field, the Flags field, and the Fragment Offset field handle fragmentation of a packet in the IPv4 header. Fragmentation happens if a large packet has to be sent over a network that only supports smaller packet sizes. In that case, the IPv4 router splits the packet into smaller slices and forwards multiple packets. The destination host collects the packets and reassembles them. If only one packet is missing or has an error, the whole transmission has to be redone; this is very inefficient. In IPv6, a host learns the Path Maximum Transmission Unit (MTU) size through a procedure called Path MTU Discovery. If a sending IPv6 host wants to fragment a packet, it will use an Extension header to do so. IPv6 routers along the path of a packet do not provide fragmentation, as they did with IPv4. So the Identification, Flags, and Fragment Offset fields were removed from the IPv6 header and will be inserted as an Extension header, if needed. Extension headers are explained later in this chapter.

Path MTU Discovery is explained in Chapter 4.

The Header Checksum field was removed to improve processing speed. If routers do not have to check and update checksums, processing becomes much faster. Checksumming is done at the media access level, too, and the risk for undetected errors and misrouted packets is minimal. There is a checksum field at the transport layer (UDP and TCP). IP is a best-effort delivery protocol; it is the responsibility of upper layer protocols to insure integrity.

The Type of Service field was replaced by the TrafficClass field. IPv6 has a different mechanism to handle preferences. Refer to Chapter 6 for more information. The Protocol Type and the Time-to-Live (TTL) fields were renamed and slightly modified. A Flow Label field was added.

2.2 The Fields in the IPv6 Header

By becoming familiar with the fields of the IPv6 header, you will better understand how IPv6 works.

For a detailed description of all the fields in an IPv4 header, refer to *Novell's Guide to Troubleshooting TCP/IP* (John Wiley & Sons) by Silvia Hagen and Stephanie Lewis.

Figure 2-1 provides an overview of the IPv6 header. The fields are discussed in detail in the following paragraphs.

Figure 2-1. Fields in the IPv6 header

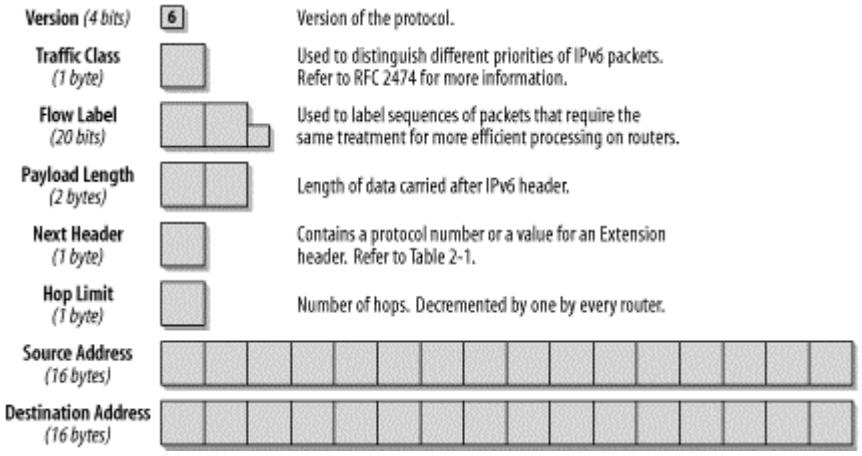


Figure 2-1 shows that even though the header has a total size of 40 bytes, which is twice as long as a default IPv4 header, it has actually been streamlined because most of the header is taken by the two 16-byte IPv6 addresses. That leaves only 8 bytes for other header information.

2.2.1 Version (4 Bits)

This is a 4-bit field and contains the version of the protocol. In the case of IPv6, the number is 6. The version number 5 could not be used because it had already been assigned an experimental stream protocol (ST2, RFC 1819).

2.2.2 Traffic Class (1 Byte)

This field replaces the Type of Service field in IPv4. This field facilitates the handling of real-time data and any other data that requires special handling. This field can be used by sending nodes and forwarding routers to identify and distinguish between different classes or priorities of IPv6 packets.

RFC 2474, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," explains how the Traffic Class field in IPv6 can be used. RFC 2474 uses the term DS Field to refer to the Type of Service field in the IPv4 header, as well as to the Traffic Class field in the IPv6 header.

2.2.3 Flow Label (20 Bits)

This field distinguishes packets that require the same treatment, in order to facilitate the handling of real-time traffic. A sending host can label sequences of packets with a set of options. Routers keep track of flows and can process packets belonging to the same flow more efficiently because they do not have to

reprocess each packet's header. A flow is uniquely identified by the flow label and the address of the source node. Nodes that do not support the functions of the Flow Label field are required to pass the field unchanged when forwarding a packet and to ignore the field when receiving a packet. All packets belonging to the same flow must have the same source and destination IP address.

The use of the Flow Label field is experimental and still under discussion at the IETF at the time of this writing. Refer to Chapter 6 for more information.

2.2.4 Payload Length (2 Bytes)

This field specifies the payload—i.e., the length of data carried after the IP header. The calculation in IPv6 is different from the one in IPv4. The Length Field in IPv4 includes the length of the IPv4 header, whereas the Payload Length field in IPv6 contains only the data following the IPv6 header. Extension headers are considered part of the payload and are therefore included in the calculation.

The fact that the Payload Length field has 2 bytes limits the maximum packet payload size to 64 KB. IPv6 has a Jumbogram Extension header, which supports bigger packet sizes, if needed. Jumbograms are relevant only when IPv6 nodes are attached to links that have a link MTU greater than 64 KB. Jumbograms are specified in RFC 2675.

2.2.5 Next Header (1 Byte)

In IPv4, this field is the Protocol Type field. It was renamed in IPv6 to reflect the new organization of IP packets. If the next header is UDP or TCP, this field will contain the same protocol numbers as in IPv4—for example, protocol number 6 for TCP or 17 for UDP. But if Extension headers are used with IPv6, this field contains the type of the next Extension header. That header is located between the IP header and the TCP or UDP header. Table 2-1 lists possible values in the Next Header field.

Table 2-1. Values in the Next Header field

Value	Description
0	In an IPv4 header: reserved and not used In an IPv6 header: Hop-by-Hop Option Header following
1	Internet Control Message Protocol (ICMPv4)—IPv4 support
2	Internet Group Management Protocol (IGMPv4)—IPv4 support
4	IP in IP (encapsulation)
6	TCP
8	Exterior Gateway Protocol (EGP)
9	IGP - any private interior gateway (used by Cisco for their IGRP)
17	UDP
41	IPv6
43	Routing header
44	Fragmentation header
45	Interdomain Routing Protocol (IDRP)
46	Resource Reservation Protocol (RSVP)
50	Encrypted Security Payload header
51	Authentication header
58	ICMPv6

59	No Next Header for IPv6
60	Destination Options header
88	EIGRP
89	OSPF
108	IP Payload Compression Protocol
115	Layer 2 Tunneling Protocol (L2TP)
132	Stream Control Transmission Protocol (SCTP)
134-254	Unassigned
255	Reserved

Header type numbers derive from the same range of numbers as protocol type numbers and should therefore not conflict with them.

The complete list of protocol numbers can be found in the appendix. For the most current list, go to IANA's web site at http://www.iana.org/assignments/protocol-numbers.

2.2.6 Hop Limit (1 Byte)

This field is analogous to the TTL field in IPv4. The TTL field contains a number of seconds, indicating how long a packet can remain in the network before being destroyed. Most routers simply decremented this value by one at each hop. This field was renamed to Hop Limit in IPv6. The value in this field now expresses a number of hops and not a number of seconds. Every forwarding node decrements the number by one.

2.2.7 Source Address (16 Bytes)

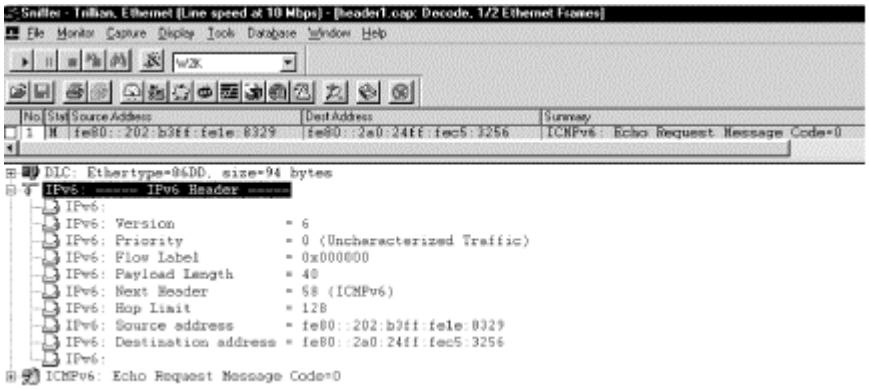
This field contains the IP address of the originator of the packet.

2.2.8 Destination Address (16 Bytes)

This field contains the IP address of the intended recipient of the packet. With IPv4, this field always contains the address of the ultimate destination of the packet. With IPv6, this field might not contain the IP address of the ultimate destination if a Routing header is present.

Figure 2-2 shows the IPv6 header in the trace file.

Figure 2-2. The IPv6 header in a trace file



This trace file shows all of the header fields I have discussed and how they are presented in a trace file. The Version field is set to 6 for IPv6. The Priority and the Flow Label fields are not used in this packet and are set to zero. The Payload Length is 40 and the Next Header value is set to 58 for ICMPv6. The Hop Limit is set to 128 and the Source and Destination addresses contain the link local addresses of my IPv6 nodes.

2.3 Extension Headers

The IPv4 header can be extended from a minimum of 20 bytes to 60 bytes in order to specify options such as Security Options, Source Routing, or Timestamping. This capacity has rarely been used because it causes a performance hit. For example, IPv4 hardware forwarding implementations have to pass the packet containing options to the main processor (software handling).

The simpler a packet header, the faster the processing. IPv6 has a new way to deal with options that has substantially improved processing. It handles options in additional headers called Extension headers.

The current IPv6 specification (RFC 2460) defines six Extension headers:

- Hop-by-Hop Options header
- Routing header
- Fragment header
- Destination Options header
- Authentication header
- Encrypted Security Payload header

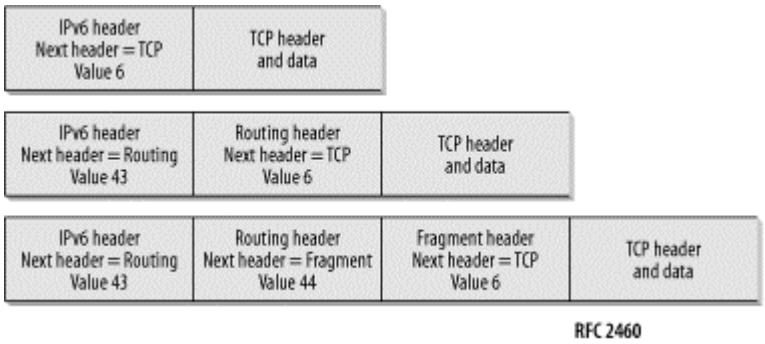
There can be zero, one, or more than one Extension header between the IPv6 header and the upper-layer protocol header. Each Extension header is identified by the Next Header field in the preceding header. The Extension headers are examined or processed only by the node identified in the Destination Address field of the IPv6 header. If the address in the Destination Address field is a multicast address, the Extension headers are examined and processed by all the nodes belonging to that multicast group. Extension headers must be strictly processed in the order they appear in the packet header.

There is an exception to the above rule: only the destination node will process an Extension header. If the Extension header is a Hop-by-Hop Options header, the information it carries must be examined and processed by every node along the path of the packet. The Hop-by-Hop Options header, if present, must immediately follow the IPv6 header. It is indicated by the value zero in the Next Header field of the IPv6 header (see Table 2-1, earlier in this chapter).

The first four Extension headers are described in RFC 2460. The Authentication header is described in RFC 2402 and the Encrypted Security Payload header in RFC 2406.

Figure 2-3 shows how Extension headers are used.

Figure 2-3. The use of Extension headers



Each Extension header is a multiple of 8 octets long. That way, subsequent headers can always be aligned. If a node is required to process the Next Header but cannot identify the value in the Next Header field, it is required to discard the packet and send an ICMPv6 Parameter Problem message back to the source of the packet. For details on ICMPv6 messages, refer to Chapter 4.

If more than one Extension header is used in a single packet, the following header order should be used (RFC 2460):

1. IPv6 header
2. Hop-by-Hop Options header
3. Destination Options header (for options to be processed by the first destination that appears in the IPv6 Destination address field, plus subsequent destinations listed in the Routing header)
4. Routing header
5. Fragment header
6. Authentication header
7. Encapsulating Security Payload header
8. Destination Options header (for options to be processed only by the final destination of the packet)
9. Upper-Layer header

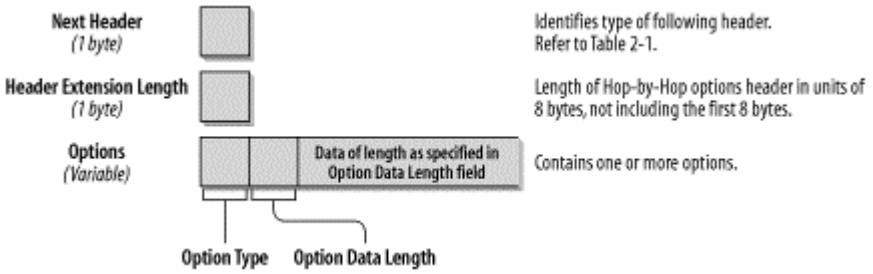
In cases when IPv6 is encapsulated in IPv4, the Upper-Layer header can be another IPv6 header and can contain Extension headers that have to follow the same rules.

2.3.1 Hop-by-Hop Options Header

The Hop-by-Hop Options Extension header carries optional information that must be examined by every node along the path of the packet. It must follow the IPv6 header immediately and is indicated by a Next Header value of zero. For example, the Router Alert (RFC 2711) uses the Hop-by-Hop Extension header for protocols like Resource Reservation Protocol (RSVP) or Multicast Listener Discovery (MLD) messages. With IPv4, the only way for a router to determine if it needs to examine a datagram is to, at least partially, parse upper layer data in all datagrams. This slows down the routing process substantially. With IPv6, in the absence of a Hop-by-Hop Extension header, a router knows that it does not need to process router-specific information and can route the packet immediately to the final destination. If there is a Hop-by-Hop Extension header, the router only needs to examine this header and not look further into the packet.

The format of the Hop-by-Hop Options header is shown in Figure 2-4.

Figure 2-4. Format of the Hop-by-Hop Options header



The following list describes each field:

Next Header (1 byte)

The Next Header field identifies the type of header that follows the Hop-by-Hop Options header. The Next Header field uses the values listed in Table 2-1, earlier in this chapter.

Header Extension Length (1 byte)

This field identifies the length of the Hop-by-Hop Options header in 8-byte units. The length calculation does not include the first 8 bytes.

Options (variable size)

There can be one or more options. The length of the options is variable and determined in the Header Extension Length field.

The Option Type Field, the first byte of the Options fields, contains information about how this option must be treated in case the processing node does not recognize the option. The value of the first two bits specifies the actions to be taken:

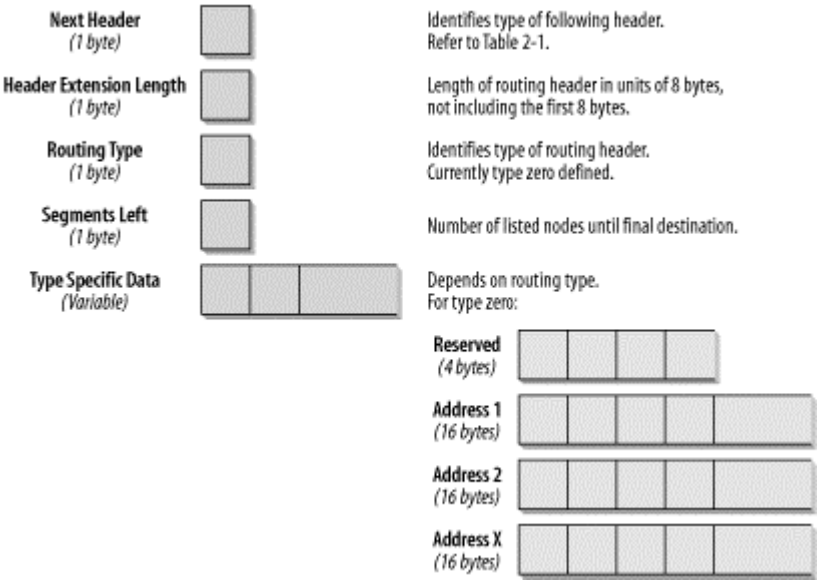
- Value 00: skip and continue processing.
- Value 01: discard the packet.
- Value 10: discard the packet and send ICMP Parameter Problem, Code 2 message to the packet's source address, pointing to the unrecognized option type.
- Value 11: discard the packet and send ICMP Parameter Problem, Code 2 message to the packet's source address only if the destination is not a multicast address.

The third bit of the Options Type field specifies whether the option information can change en route (value 01) or does not change en route (value 00).

2.3.2 Routing Header

The Routing header is used to give a list of one or more intermediate nodes that should be visited on the packet's path to its destination. In the IPv4 world, this is called the Loose Source and Record Route option. The Routing header is identified by a Next Header value of 43 in the immediately preceding header. Figure 2-5 shows the format of the Routing header.

Figure 2-5. Format of the Routing header



The following list describes each field:

Next Header (1 byte)

The Next Header field identifies the type of header that follows the Routing header. It uses the same values as the IPv4 Protocol Type field (see Table 2-1, earlier in this chapter).

Header Extension Length (1 byte)

This field identifies the length of the Routing header in 8-byte units. The length calculation does not include the first 8 bytes.

Routing Type (1 byte)

This field identifies the type of Routing header. RFC 2460 describes Routing Type zero.

Segments Left (1 byte)

This field identifies how many nodes are left to be visited before the packet reaches its final destination.

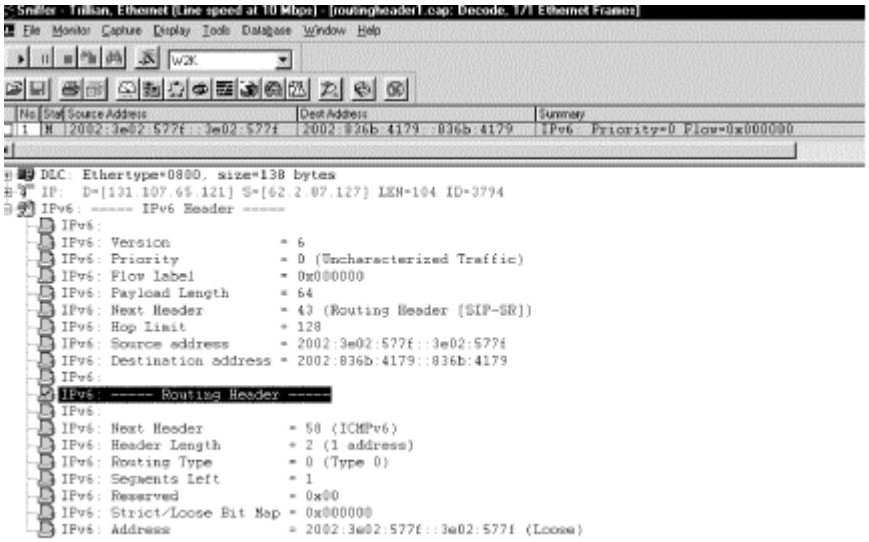
Type-Specific Data (Variable-length)

The length of this field depends on the Routing Type. The length will always make sure that this complete header is a multiple of 8 bytes.

If a node processing a Routing header cannot identify a Routing Type value, the action taken depends on the content of the Segments Left field. If the Segments Left field does not contain any nodes to be visited, the node must ignore the Routing header and process the next header in the packet, determined by the Next Header field value. If the Segments Left field is not zero, the node must discard the packet and send an ICMP Parameter Problem, Code 0 message to the packet's source address, pointing to the unrecognized Routing Type. If a forwarding node cannot process the packet because the next link MTU size is too small, it discards the packet and sends an ICMP Packet Too Big message back to the source of the packet.

The only Routing Type described in RFC 2460 is a Type Zero Routing header. The first node that processes the Routing header is the node addressed by the Destination address field in the IPv6 header. This node decrements the Segments Left field by one and inserts the next address field from within the Routing header in the IPv6 header Destination address field. Then the packet is forwarded to the next hop that will again process the Routing header as described until the final destination is reached. The final destination is the last address in the Routing Header Data field. For example, Mobile IPv6 uses the Routing header. Any node sending a packet to a mobile node will send the packet to the mobile node's care-of-address. It will include a Routing header with one entry, the mobile node's home address. The mobile node swaps the Destination address in the IPv6 header with the entry in the Routing header and will reply with its home address as a source address as if it received the packet attached to its home network. For further discussion and definition of terms regarding Mobile IPv6, refer to Chapter 7. Figure 2-6 shows the routing header in a trace file.

Figure 2-6. Routing header in a trace file



The Next Header field within the IPv6 header shows the value 43 for the Routing header. The Source and Destination addresses have the prefix 2002:, which is allocated to 6to4 sites. The Routing header contains the fields discussed earlier in this section. Next Header will be ICMPv6, value 58. The Header Length is two 8-byte units, which calculates to a total length of 16 bytes. The Segments Left field contains the value 1 because there is one address entry in the Options Fields. Finally, the Options field lists the addresses to be visited. In this case, there is only one entry. If a number of hosts is listed here, every forwarding node (that is, the destination IP address in the IPv6 header) takes the next entry from this host list, uses it as a new destination IP address in the IPv6 header, decrements the Segments Left field by one, and forwards the packet. This is done until the last host in the list is reached. RFC 2460 shows an example.

A source node S sends a packet to destination node D using a Routing header to send the packet through the intermediate nodes I1, I2, and I3. The Routing header changes are shown in Table 2-2.

Table 2-2. Processing the Routing header

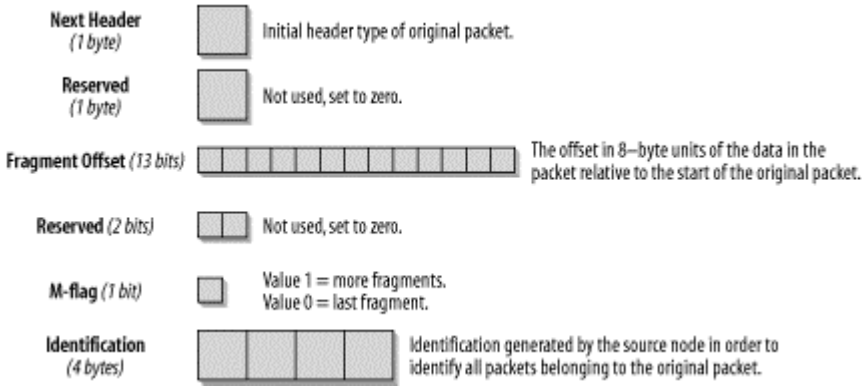
	IPv6 Header	Routing Header
Packet from S to I1	Source address S Destination address I1	Segments Left 3 Address (1) = I2 Address (2) = I3 Address (3) = D

Packet from I1 to I2	Source address S Destination address I2	Segments Left 2 Address (1) = I1 Address (2) = I3 Address (3) = D
Packet from I2 to I3	Source address S Destination address I3	Segments Left = 1 Address (1) = I1 Address (2) = I2 Address (3) = D
Packet from I3 to D	Source address S Destination address D	Segments Left = 0 Address (1) = I1 Address (2) = I2 Address (3) = I3

2.3.3 Fragment Header

An IPv6 host that wants to send a packet to an IPv6 destination uses Path MTU discovery to determine the maximum packet size that can be used on the path to that destination. If the packet to be sent is larger than the supported MTU, the source host fragments the packet. Unlike IPv4, with IPv6, a packet does not get fragmented by a router along the path. Fragmentation only occurs on the source host sending the packet. The destination host handles reassembly. A Fragment header is identified by a Next Header value of 44 in the preceding header. The format of the Fragment header is shown in Figure 2-7.

Figure 2-7. Format of the Fragment header



The following list describes each field:

Next Header (1 byte)

The Next Header field identifies the type of header that follows the Fragment header. It uses the same values as the IPv4 Protocol Type field. (See Table 2-1).

Reserved (1 byte)

Not used; set to zero.

Fragment Offset (13 bits)

The offset in 8-byte units of the data in this packet relative to the start of the data in the original packet.

Reserved (2 bits)

Not used; set to zero.

M-Flag (1 bit)

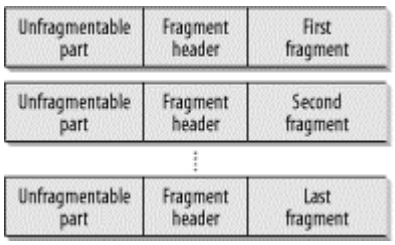
Value 1 indicates more fragments; value zero indicates last fragment.

Identification (4 Bytes)

Generated by the source host in order to identify all packets belonging to the original packet. This field is usually implemented as a counter, increasing by one for every packet that needs to be fragmented by the source host.

The initial unfragmented packet is referred to as the original packet. It has an unfragmentable part that consists of the IPv6 header, plus any Extension headers that must be processed by nodes along the path to the destination (i.e., Hop-by-Hop Options). The fragmentable part of the original packet consists of any Extension headers that need only to be processed by the final destination, plus the Upper-Layer headers and any data. Figure 2-8 (RFC 2460) illustrates the fragmenting process.

Figure 2-8. Fragmentation with IPv6

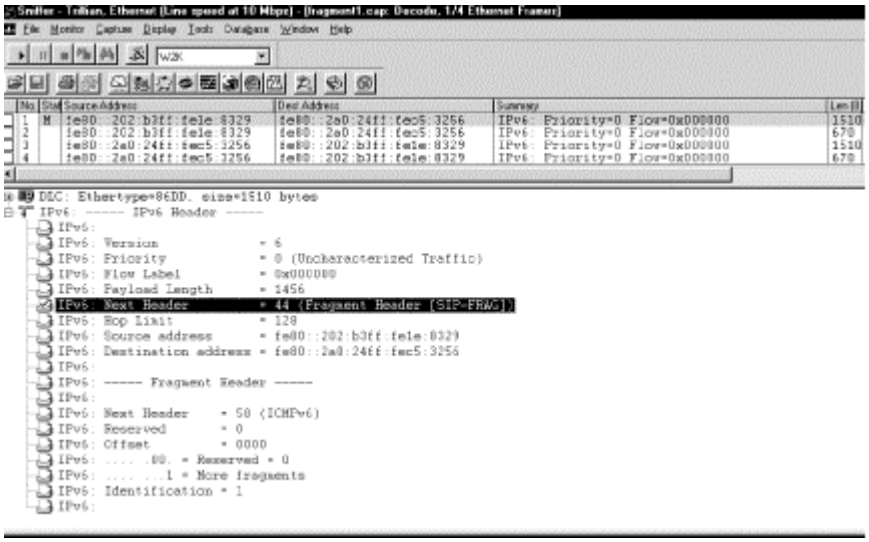


The unfragmentable part of the original packet appears in every fragment, followed by the Fragmentation header, and then the fragmentable data. The IPv6 header of the original packet has to be slightly modified. The length field reflects the length of the fragment (excluding the IPv6 header) and not the length of the original packet.

The destination node collects all the fragments and reassembles them. The fragments must have identical Source and Destination addresses and the same identification value in order to be reassembled. If all fragments do not arrive at the destination within 60 seconds after the first fragment, the destination will discard all packets. If the destination has received the first fragment (offset = zero), it sends back an ICMPv6 Fragment Reassembly Time Exceeded message to the source.

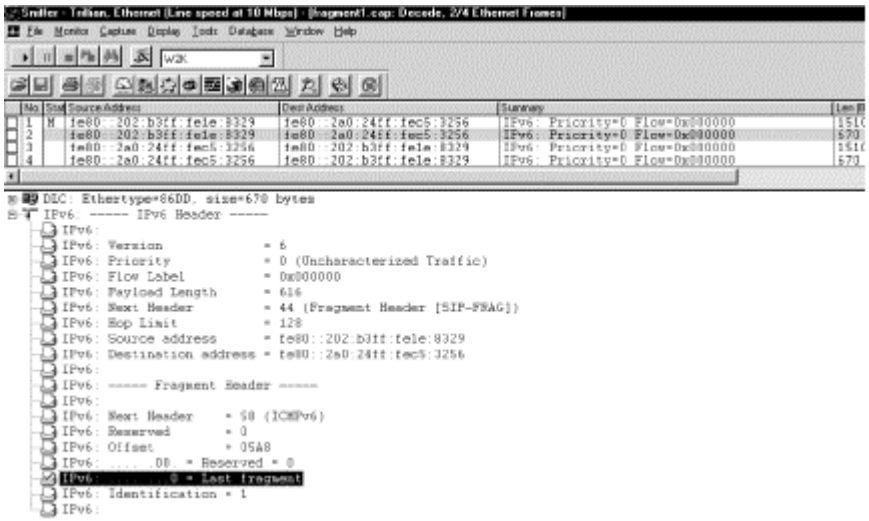
Figure 2-9 shows a Fragment header.

Figure 2-9. Fragment header in a trace file



I created this Fragment header by generating an oversized ping from *Marvin* to *Ford* (Win2000 to Linux). The whole fragment set consists of two packets, the first of which is shown in Figure 2-9. In the IPv6 header, the Payload Length field has a value of `1456`, which is the length of the fragmentation header and this one fragment, not the length of the whole original packet. The Next Header field specifies the value `44`, which is the value for the Fragment header. This field is followed by the Hop Limit field and by the Source and Destination IP addresses. The first field in the Fragment header is the Next Header field. Because this is a ping, it contains the value `58` for ICMPv6. And because this is the first packet in the fragment set, the value in the Offset field is zero and the M-Flag is set to one, which means there are more fragments to come. The Identification field is set to one and has to be identical in all packets belonging to this fragment set. Figure 2-10 shows the second packet of the fragment set.

Figure 2-10. The last packet in the fragment set

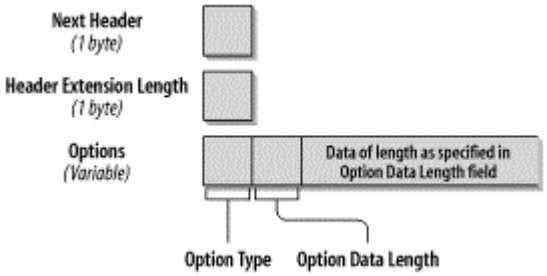


The second and last packet of this fragment set has an Offset value of `0x05A8`, which translates to 1448 in decimal, the length of the first fragment. The M-Flag is set to zero. This indicates that it is the last packet and tells the receiving host that it is time to reassemble the fragments. The Identification field is set to one in both packets.

2.3.4 Destination Options Header

A Destination Options header carries optional information that is examined by the destination node only. The Next Header value identifiying this type of header is the value `60`. Figure 2-11 shows the format of the Destination Options header.

Figure 2-11. Format of the Destination Options header



The following list describes each field:

Next Header (1 byte)

> The Next Header field identifies the type of header that follows the Destination Options header. It uses the same values listed in Table 2-1, earlier in this chapter.

Header Extension Length (1 byte)

> This field identifies the length of the Destination Options header in 8-byte units. The length calculation does not include the first 8 bytes.

Options (variable size)

> There can be one or more options. The length of the options is variable and determined in the Header Extension Length field.

The Options field is used in the same way as the Hop-by-Hop Options header, which I discussed earlier in this chapter. An example of the Destination Options header is Mobile IPv6. A mobile IPv6 node connected to a foreign network can send packets with its care-of-address as a source address and its home address in a home address destination option. According to the current Mobile IPv6 draft, the ability to correctly process a home address in a Destination Option is required in all IPv6 nodes. For a detailed explanation of Mobile IPv6, refer to Chapter 7 or to the current draft of Mobile IPv6 at http://www.ietf.org/internet-drafts/draft-ietf-mobileip-ipv6-18.txt. Note that the draft number may have increased by one or more when you follow this link.

Chapter 3. IPv6 Addressing

An IPv4 address has 32 bits and is familiar. An IPv6 address has 128 bits and looks wild. Extending the address space was one of the driving reasons to develop IPv6, along with optimization of routing tables, especially on the Internet. This chapter will help you become familiar with the extended address space and will also explain how IPv6 addressing works and why it has been designed the way it is. The IPv6 addressing architecture is defined in RFC 2373, which obsoletes RFC 1884.

3.1 Address Types

IPv4 knows unicast, broadcast, and multicast addresses. With IPv6, the broadcast address is not used anymore; multicast addresses are used instead. This is good news because broadcasts are a problem in most networks. The *anycast* address, a new type of address introduced with RFC 1546, is now used with IPv6.

3.1.1 Unicast, Multicast, and Anycast Addresses

An IPv6 address can be classified into one of three categories:

Unicast

A unicast address uniquely identifies an interface of an IPv6 node. A packet sent to a unicast address is delivered to the interface identified by that address.

Multicast

A multicast address identifies a group of IPv6 interfaces. A packet sent to a multicast address is processed by all members of the multicast group.

Anycast

An anycast address is assigned to multiple interfaces (usually on multiple nodes). A packet sent to an anycast address is delivered to only one of these interfaces, usually the nearest one.

3.1.2 Some General Rules

IPv6 addresses are assigned to interfaces, as in IPv4, not to nodes, as in OSI, so each interface of a node needs at least one unicast address. A single interface can also be assigned multiple IPv6 addresses of any type (unicast, multicast, anycast). A node can therefore be identified by the address of any of its interfaces. It is also possible to assign one unicast address to multiple interfaces for load-sharing reasons, but if you do this, you need to make sure that the hardware and the drivers support it. With IPv6, all zeros and ones are legal values for any field in an address.

A typical IPv6 address consists of three parts—the *global routing prefix*, the *subnet ID,* and the i*nterface ID*—as shown in Figure 3-1.